Research Paper | Computer Science | E-ISSN No : 2454-9916 | Volume : 6 | Issue : 9 | Sep 2020

KEYWORD SEARCH-BASED DATA INTEGRATION BY NEW SOURCES

Bhanu Shanker Prasad

P G, Deptt of Statistics and Computer Application, TMBU, Bhagalpur, Bihar, India.

ABSTRACT

Now a days scientific data offers some of the most interesting challenges in data integration. Scientific field evolve rapidly and accumulate masses of observational and experimental data that needs to be annotated, revised interlinked and made available to other scientists. From the user point of view, this can be major headache as the data they seek may initially be spread across many databases in need of integration. The purpose of this paper is to present recent ideas for creating integrated views over data sources using keyword search techniques, ranked answers and user feedback to investigate how to automatically discover when a new data source has content relevant to a user's view – in essence, performing automatic data integration for incoming data sets.

KEYWORDS: User feedback, data integration, keyword search, data sets.

INTRODUCTION:

Data integration remains one of the most difficult challenges in information technology, largely due to the ambiguities involved in trying to semantically merge different data sources. In an ideal world, the data needs of science, medicine, and policy would be met by discovering new data sets and databases the moment they are published, and automatically conveying their contents to users with related information needs, in the form relevant to those users. Instead, we live in a world where both discovery and semantic conversion are for the most part time-consuming, manual processes, causing a great deal of relevant information to be simply ignored. To address these difficulties, some research communities have attempted to define a consensus global schema (mediated schema) for their field so that individual sources can be mapped into a common representation. Researchers in machine learning, databases, and the semantic web have made significant progress in recent years on partially automating these mapping and alignment tasks [29].

However, the global-schema approach is poorly suited to automating the process of source discovery and integration in a dynamic scientific community. It is difficult to develop a consensus mediated schema that captures the diverse needs of a large user base and keeps up with new concepts, methods, and types of experimental result. Few mechanisms exist for discovering relevant sources as they are first published, and for having their data automatically put into use. Finally, schema alignment tools rarely scale to large numbers of schemas and relations, and it can be difficult to determine when they have produced the right mappings.

The work on the Q system [32] develops an information need- driven paradigm for data integration, which addresses the above problems. Q is initially given a set of databases that contain known cross-references, links, and correspondence or cross-reference tables; it does not require a global mediated schema or full schema mappings. A user specifies an information need through a keyword query. Leveraging ideas from keyword search in databases [4, 5, 17, 20], Q defines a ranked view consisting of a union of conjunctive queries over different combinations of the sources. This view is made persistent and refined through user feedback.

Here, we build upon Q's information need-driven integration model by addressing the challenge of automatically adding new data sources and relating them to the existing ones. As a user (or a Web crawler) registers a new database, that source's relevance to existing ranked views is considered, using information about data-value overlap as well as schema alignment costs from existing schema matchers. Going beyond our previous work [32], Q can now combine the weighted outputs from different schema matchers. If the source is found to be highly relevant to a ranked view, then query results are refreshed as appropriate. Now the users of the view may provide feedback on its contents: certain new results may be valuable, or possibly erroneous. As the system gets feedback about erroneous results, it adjusts the costs it has assigned to specific mappings or alignments so that associations responsible for the errors are avoided. Q can adjust weights for individual alignments, including how much to favor the outputs from different schema matchers.

Our work distinguishes itself from prior efforts in interactive, user-driven integration (e.g., dataspaces [14] and best-effort integration [30]) by automatically discovering semantic links among data sources, and using a data-driven approach to providing feedback to the system. Any form of automatic schema alignment is likely to make errors, especially at scale; the challenge is to determine when and where there are mistakes. Simply "eyeballing" the output mapping is unlikely to help identify what is correct. However, if a domain expert is looking at data from the perspective of a particular information need, he or she is (1) likely to invest some effort in ensuring the quality of results, (2) likely to recognize when results do not make sense.

Search-Based Integration:

Here, we adopt a keyword search query model [4, 17, 20, 32] in which keywords are matched against elements in one or more relations in different data sources. The system attempts to find links between the relations matching the given keywords. Such links are proposed by different kinds of associations such as foreign key relationships, value overlaps or global identifiers, similarity predicates, or hyperlinks. In general, there may be multiple relations matching a search keyword, and multiple attribute pairs may align between relations, suggesting many possible ways to join relations in order to answer the query.

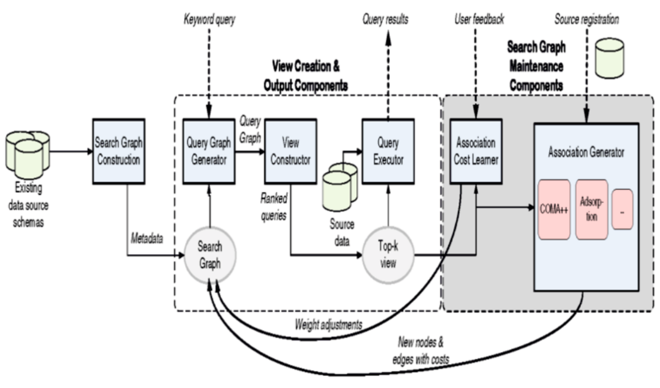

Figure 1: Basic architecture of Q. The initial search graph comes from the sources known at startup. At query time this is expanded into a query graph, from which queries and ultimately results are generated. The *search graph maintenance* modules, the focus of this paper, handle user feedback and accept new source registrations, in order to update the search graph with new alignments — triggering recomputation of the query graph and query results in response.

Figure 1 shows the basic architecture of our Q system. We start with an initial search graph generated from existing data source relations and the associations among them. During the view creation and output stage, a keyword search is posed against this search graph, and results in a top-k view containing answers believed to be relevant to the user. The definition and contents of this view are maintained continuously: both the top-scoring queries and their results may need to be updated in response to changes to the underlying search graph made (1) directly by the user, who may provide feedback that changes the costs of certain queries and thus query answers; (2) by the system, as new data sources are discovered, and their attributes are found to align with the existing relations in the search graph, in a way that results in new top-fc answers for the user's view. We refer to the process of updating the schema graph's nodes and associations as search graph maintenance. In fact, there is interplay between the two graph maintenance mechanisms and the view creation and output stage, as the system may propose an alignment, the view's contents may be updated, the user may provide feedback on these results, and the view output may be updated once again. All of this is focused around alignments that are relevant to the user's ongoing information need.

Adding New Data Sources:
Once a keyword search-based view has been defined as in the previous section, Q switches into search graph maintenance mode. One crucial maintenance process, discussed in this section, decides if and how to incorporate new sources into the current view as the system is notified of their availability.

Q includes a registration service for new tables and data sources: this mechanism can be manually activated by the user (who may give a URL to a remote JDBC source), or could ultimately be triggered directly by a Web crawler that looks for and extracts tables from the Web [7] or the deep Web [24, 35] .

Basic Approach:
When a new source is registered, the first step is to incorporate each of its underlying tables into the search graph. The search graph is in effect the data model queried by Q. It contains both metadata (relation and attribute nodes) and data (tuple values), related by edges that specify possible ways of constructing a query. The lower the cost of an edge, the more likely that the edge will be relevant to answering queries involving one of the nodes it links.

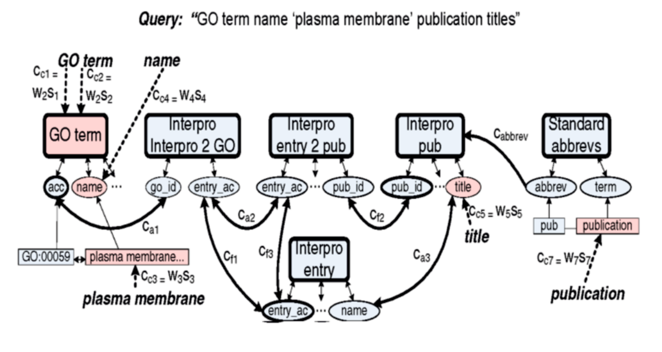

Figure 3: Query graph, given a series of keyword search terms. In general, each keyword may match a node with a *similarity score* s_{ci}, for which a *weight coefficient* w_{ci} is to be assigned by the system.

When a new source is encountered, the first step is to determine potential alignments between the new source's attributes and those in existing tables: these alignments will suggest (1) potential joins to be used in query answering, and (2) potential alignments of attributes in query output, such that the same column in the query answers contains results from different sources. We note that in both cases, it is desirable that aligned attributes come from the same domains (since, in the case of joins, no results would be produced unless there are shared data values among the attributes).

Of course, this task requires a set of alignment primitives (schema matching algorithms) used to match attributes, which we describe . But there are additional architectural challenges that must be faced at the overall system level. As the search graph grows in size, the cost of adding new associations becomes increasingly expensive: regardless of the specific primitives used, the cost of alignment tends to be at least quadratic in the number of compatible attributes. We must find ways of reducing the space of possible alignments considered. Moreover, not all of these proposed alignments may be good ones: most schema matching or alignment algorithms produce false positives.

We exploit the fact that a bad alignment will become apparent when (and only when) it affects the top-fc results of a user query whose results are closely inspected. We develop an information need-driven strategy where we consider only alignments that have the potential to affect existing user queries . As we later show, this restricts the space of potential alignments to a small subset of the search graph, which grows at a much lower rate than the search graph itself. We then develop techniques for correcting bad alignments through user feedback on the results of their queries .

Experimental Analysis:
In this section, we use Q as a platform to validate our strategy of performing schema alignment in a query-guided manner as well as our techniques for using user feedback over data to correct bad alignments . The search graph maintenance modules in Q comprise approximately 4000 lines of Java code, and all experiments were run on a Dell PowerEdge 1950 computer running RedHat Enterprise Linux 5.1 with 8GB RAM. We used the COMA++ 2008 API, and a Java-based implementation of our MAD-based schema matcher.

Our focus in Q is on supporting bioinformatics applications, and hence wherever possible, we use real biological databases and compare with gold standard results, i.e., reference results supplied by domain experts. This enables us to perform an experimental study without having to conduct extensive user studies.

For the first set of experiments, we use a dataset for which we have logs of actual SQL queries executed by Web forms, such that we can determine which proposed source associations are actually valid (as witnessed by having real queries use them). This dataset, GBCO, consists of 18 relations (which we model as separate sources) with 187 attributes.

In the second set of experiments, we used a different dataset, based on the widely used (and linked) Interpro and GO databases, where we could obtain keyword queries and find multiple alternative means of answering these queries. This dataset consists of 8 closely interlinked tables with 28 attributes.

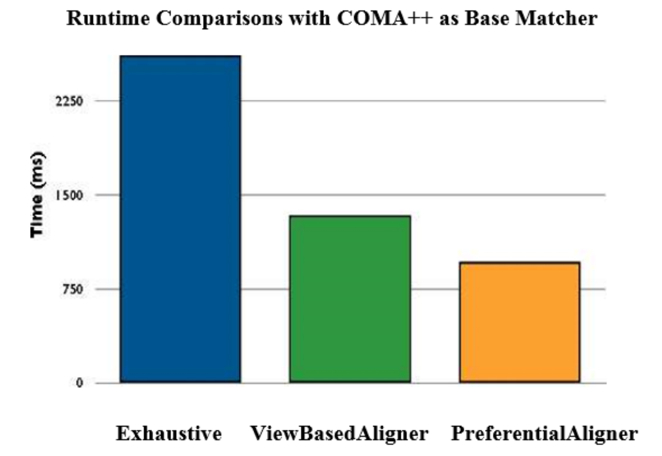

Figure 6: Running times (avgd. over intro of 40 sources) when aligning a new source to a set of existing sources (COMA++ as base matcher). VIEWBASEDALIGNER and PREFERENTIAL-ALIGNER significantly reduce running times vs. EXHAUSTIVE.

Incorporating New Sources:
We first look at the cost of adding new data sources to an existing search graph, in a way that keeps the alignment task tractable by limiting it to the "neighborhood" of an existing query. We set up the experiment, using the GBCO dataset described above, as follows.

We first scanned through the GBCO query logs for pairs of SQL queries, where one query represented an expansion of the other, base, query: i.e., the expanded query either joined or unioned additional relations with the base query. Intuitively, the expanded query tells us about new sources that would be useful to add to an existing search graph that had been capable of answering the base query. When the expanded query represents the union of the base query with a new query sub expression, then clearly adding the new data source results in new association edges that provide further data for the user's view. When the expanded query represents an additional join of the base query with new data, this also affects the contents of the existing view if the additional join represents a segment of a new top-scoring Steiner tree for the same keyword query.

For each base query, we constructed a corresponding keyword query, whose Steiner trees included the relations in the base query. Next, we initialized the search graph to include all sources except the ones unique to the expanded query. We initially set the weights in the search graph to default values, then provided feedback on the keyword query results, such that the SQL base query from our GBCO logs was returned as the top query. For all experiments in this section, the edge costs learned in the process were used as the value of the function C in the VIEWBASEDALIGNER algorithm. The vertex cost function P in PREFERENTIALALIGNER was similarly estimated from the weights of features corresponding to source identities.

Cost of Alignment:
Our first experiment measures the cost of performing alignments between the new source and a schema graph containing all of the other sources — using our EXHAUSTIVE, VIEWBASEDALIGNER, and PREFERENTIALALIGNER search strategies, with the COMA++ matcher. Figure 6 compares the running times of these strategies. Figure 7 shows the number of pair wise attribute comparisons necessary, under two different sets of assumptions. The Value Overlap Filter case assumes we have a content index available on the attributes in the existing set of sources and in the new source; we only make compare attributes that have shared values (hence can join). More representative is likely to be the No Additional Filter case, which has only metadata to work from.

We observe that, regardless of whether a value overlap filter is available, limiting the search to the neighborhood of the existing query (i.e., our information need-driven pruning strategy) provides significant speedups (about 60%) versus doing an exhaustive set of comparisons, even on a search graph that is not huge. Recall that VIEWBASEDALIGNER will provide the exact same updates to a user view as the exhaustive algorithm. PREFERENTIALALIGNER does not have this guarantee, and instead focuses on the alignments specified in the prior, but gives even lower costs.

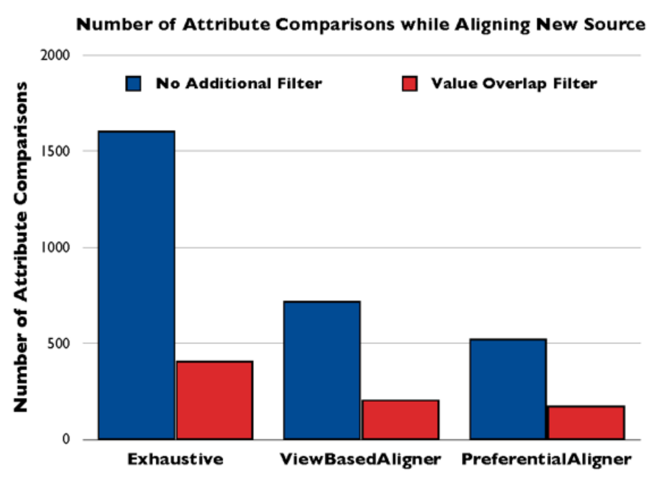

Figure 7: Pairwise attribute comparisons performed in aligning new source(s) to existing sources (avgd. over intro of 40 sources in 16 trials, where each trial introduces one or more new sources). VIEWBASEDALIGNER and PREFERENTIALALIGNER significantly reduce comparisons vs. EXHAUSTIVE.

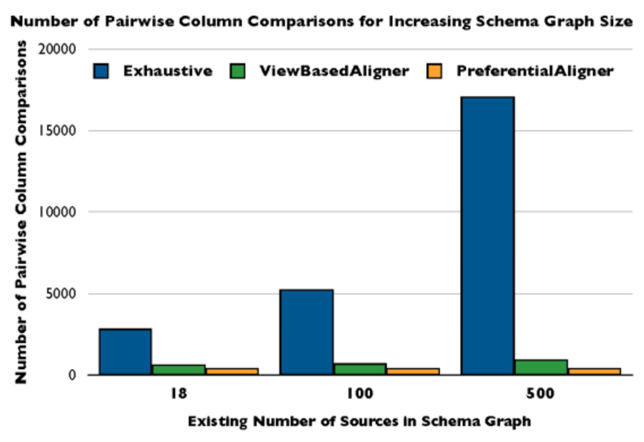

Figure 8: Number of pairwise attribute comparisons as the size of the search graph is increased (avgd. over introduction of 40 sources). VIEWBASEDALIGNER and PREFERENTIALALIGNER are hardly affected by graph size.

The differences in costs results from the fact that the number of comparisons in EXHAUSTIVE depends on the number of source relations in the schema graph, whereas the number of comparisons in the other cases is only dependent on the number of nodes in the local neighborhood of the query.

CONCLUSIONS AND FUTURE WORK:

Here, we have developed an automatic, information need- driven strategy for automatically incorporating new sources and their information in a data integration setting. Schema matches or alignments, whether good or bad, only become apparent when they are used to produce query answers seen by a user; we exploit this to make the process of finding alignments with a new source more efficient, and also to allow the user with an information need to actually correct bad mappings through explicit feedback (from which the system learns new association weights). Through experiments on real-world datasets from the bioinformatics domain, we have demonstrated that our alignment scheme scales well. We have also demonstrated that our learning strategy is highly effective in combining the outputs of "black box" schema matchers and in re- weighting bad alignments. In doing this, we have also developed a new instance-based schema matcher using the MAD algorithm.

We believe that Q represents a step towards the ultimate goal of automated data integration, at least for particular kinds of datasets. In ongoing work we are expanding our experimental study to consider a wider array of domains, including Web sources with information extraction components.

REFERENCES:

I. A. Balmin, V. Hristidis, and Y. Papakonstantinou. ObjectRank: Authority-based keyword search in databases. In VLDB, 2004.

II. S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for YouTube: taking random walks through the view graph. In WWW. ACM New York, NY, USA, 2008.

III. A. Barabasi and R. Albert. Emergence of scaling in random networks. Science, 286(5439):509, 1999.

IV. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In ICDE, 2002.

V. C. Botev and J. Shanmugasundaram. Context-sensitive keyword search and ranking for XML. In WebDB, 2005.

VI. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. Computer Networks, 30(1-7), 1998.

VII. M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. VLDB, 2008.

VIII. X. Chai, B.-Q. Vuong, A. Doan, and J. F. Naughton. Efficiently incorporating user feedback into information extraction and integration programs. In SIGMOD, New York, NY, USA, 2009.

IX. W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In SIGMOD, 1998.

X. K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. Journal of Machine Learning Research, 7:551-585, 2006.

XI. H. H. Do and E. Rahm. Matching large schemas: Approaches and evaluation. Inf. Syst., 32(6), 2007.

XII. A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In SIGMOD, 2001.

XIII. X. L. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In VLDB, 2007.

XIV. M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. SIGMOD Rec., 34(4), 2005.

XV. L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an RDBMS for web data integration. In WWW, 2003.

XVI. H. He, H. Wang, J. Yang, and P. S. Yu. Blinks: ranked keyword searches on graphs. In SIGMOD, 2007.